# Ponteiros e alocação de memória (supor ambiente Linux)

Analise o código abaixo e marque V para as assertivas verdadeiras e F para as falsas. Justificar as falsas.

Versão 01

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define TAM 3
4
5  void leia_vetor(int *v) {
6       int     i;
7
8       for (i=0; i<TAM; i++) {
9        printf("v[%d] : ", i);
10       scanf("%d", v[i]);
11      } /* fim-for */
12
13 } /* fim-leia_vetor() */
14
15 int main (void) {
16      int v[TAM];
17      int i;
18      leia_vetor(v);
19      for (i=0; i<TAM; i++) {
20       printf(" %d ", *(v+i));
21      } /* fim-for */
22      printf("\n");
23      return (0);
24 } /* fim-main */
```

Versão 02

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define TAM 3
4
5  int * leia_vetor() {
6       int   *n;
7
8       n = (int *) malloc (sizeof(int));
9       scanf("%d", n);
10      return (n);
11 } /* fim-leia_vetor() */
12
13 int main () {
14      int *v[TAM];
15      int i;
16
17      for (i=0; i<TAM; i++) {
18            printf("v[%d] = ", i);
19            v[i] = leia_vetor();
20      } /* fim-for */
21
22      for (i=0; i<TAM; i++) {
23            printf(" %d ", *(v[i]));
24      } /*fim-for */
25      printf("\n");
26      return (0);
27 } /* fim-main */
```

a) ( ) Do jeito que está apresentado, o programa versão 01 acima não funciona, já que a função `main` não terá acesso aos valores preenchidos em `v` pela `leia_vetor` (linha 5). Isso ocorre porque a função chamada recebe parâmetro por valor e os valores preenchidos nessa função não estão sendo retornados (como cópia ou como referência de endereço) para a função chamadora

b) ( ) Na versão 01, a variável `v` usada como parâmetro da função `leia_vetor` nas linhas 5 e 18 estão alocadas na mesma posição de memória (ou seja, não são variáveis distintas).

c) ( ) O programa versão 01 acima possui um erro de sintaxe na linha 10, pois o `scanf` precisa do endereço de uma variável e não de um conteúdo (ou seja, o comando na linha 10 deveria ser `scanf("%d", &v[i];`)

d) ( ) O programa versão 01 acima possui um erro de execução na passagem de parâmetros envolvendo as linhas 18 e 5. Ou seja, a função chamada `leia_vetor()` aguarda um apontador e a chamada para essa função está sendo feita com um parâmetro que não é um apontador (variável `v`)

e) ( ) No programa versão 01 acima, uma única alteração na linha 10 pelo comando `scanf("%d", v+i);` não causará prejuízo de compilação e nem de execução. Nesse caso, `v` será incrementado de uma unidade a cada passada do laço (considerar que um `int` ocupa 4 bytes na memória)

f) ( ) No programa versão 01 acima, uma única alteração na linha 5 substituindo o parâmetro da função por `int v[]` não causará prejuízo de compilação e nem de execução

g) ( ) A declaração feita na versão 02/linha 14, provoca alocação estática de um vetor cujas posições são preenchidas com endereços de memória que serão preenchidos com números inteiros

h) ( ) Alocações dinâmicas de memória feitas no corpo de uma função não podem ser utilizadas pela função chamadora, já que as variáveis declaradas e alocações de memória feitas no corpo dessa função chamada são eliminadas, tão logo ela finalize. Esse é um fator que inviabiliza o funcionamento adequado da versão 02 do programa acima

i) ( ) Na versão 02, linha 23, o programa voltará a funcionar normalmente se for substituído o comando de impressão `printf(" %d ", *(v[i]))` por `printf(" %d ", v[i])`

j) ( ) Na versão 02, linha 23, o programa não funcionará se o comando `printf(" %d ", *(v[i]))` for substituído por `printf(" %d ", **(v+i))`